

- [42] B. Baudry and M. Monperrus, “The multiple facets of software diversity: Recent developments in year 2000 and beyond,” *ACM Comput. Surv.*, vol. 48, sep 2015.
- [43] K. Pohl, G. Böckle, and F. Van Der Linden, *Software product line engineering: foundations, principles, and techniques*, vol. 1. Springer, 2005.
- [44] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. H. Tse, “Adaptive random testing: The art of test case diversity,” *J. Syst. Softw.*, vol. 83, pp. 60–66, 2010.
- [45] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, “Managing performance vs. accuracy trade-offs with loop perforation,” in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ESEC/FSE ’11*, (New York, NY, USA), p. 124–134, Association for Computing Machinery, 2011.
- [46] Avizienis and Kelly, “Fault tolerance by design diversity: Concepts and experiments,” *Computer*, vol. 17, no. 8, pp. 67–80, 1984.
- [47] F. B. Cohen, “Operating system protection through program evolution,” *Computers & Security*, vol. 12, no. 6, pp. 565–584, 1993.
- [48] A. Avizienis, “The methodology of n-version programming,” *Software fault tolerance*, vol. 3, pp. 23–46, 1995.
- [49] B. Randell, “System structure for software fault tolerance,” *SIGPLAN Not.*, vol. 10, p. 437–449, apr 1975.
- [50] S. Forrest, A. Somayaji, and D. Ackley, “Building diverse computer systems,” in *Proceedings. The Sixth Workshop on Hot Topics in Operating Systems (Cat. No.97TB100133)*, pp. 67–72, 1997.
- [51] T. Jackson, *On the Design, Implications, and Effects of Implementing Software Diversity for Security*. PhD thesis, University of California, Irvine, 2012.
- [52] J. V. Cleemput, B. Coppens, and B. De Sutter, “Compiler mitigations for time attacks on modern x86 processors,” *ACM Trans. Archit. Code Optim.*, vol. 8, jan 2012.
- [53] A. Homescu, S. Neisius, P. Larsen, S. Brunthaler, and M. Franz, “Profile-guided automated software diversity,” in *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pp. 1–11, IEEE, 2013.
- [54] M. Jacob, M. H. Jakubowski, P. Naldurg, C. W. N. Saw, and R. Venkatesan, “The superdiversifier: Peephole individualization for software protection,” in *International Workshop on Security*, pp. 100–120, Springer, 2008.

- [55] R. M. Tsoupidi, R. C. Lozano, and B. Baudry, “Constraint-based software diversification for efficient mitigation of code-reuse attacks,” *ArXiv*, vol. abs/2007.08955, 2020.
- [56] S. Bhatkar, D. C. DuVarney, and R. Sekar, “Address obfuscation: an efficient approach to combat a board range of memory error exploits,” in *Proceedings of the USENIX Security Symposium*, 2003.
- [57] S. Bhatkar, R. Sekar, and D. C. DuVarney, “Efficient techniques for comprehensive protection from memory error exploits,” in *Proceedings of the USENIX Security Symposium*, pp. 271–286, 2005.
- [58] K. Pettis and R. C. Hansen, “Profile guided code positioning,” in *Proceedings of the ACM SIGPLAN 1990 Conference on Programming Language Design and Implementation*, PLDI ’90, (New York, NY, USA), p. 16–27, Association for Computing Machinery, 1990.
- [59] S. Crane, A. Homescu, S. Brunthaler, P. Larsen, and M. Franz, “Thwarting cache side-channel attacks through dynamic software diversity,” in *NDSS*, pp. 8–11, 2015.
- [60] A. Romano, D. Lehmann, M. Pradel, and W. Wang, “Wobfuscator: Obfuscating javascript malware via opportunistic translation to webassembly,” in *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP)*, (Los Alamitos, CA, USA), pp. 1101–1116, IEEE Computer Society, may 2022.
- [61] M. T. Aga and T. Austin, “Smokestack: thwarting dop attacks with runtime stack layout randomization,” in *Proc. of CGO*, pp. 26–36, 2019.
- [62] S. Lee, H. Kang, J. Jang, and B. B. Kang, “Savior: Thwarting stack-based memory safety violations by randomizing stack layout,” *IEEE Transactions on Dependable and Secure Computing*, 2021.
- [63] Y. Younan, D. Pozza, F. Piessens, and W. Joosen, “Extended protection against stack smashing attacks without performance loss,” in *2006 22nd Annual Computer Security Applications Conference (ACSAC’06)*, pp. 429–438, 2006.
- [64] Y. Xu, Y. Solihin, and X. Shen, “Merr: Improving security of persistent memory objects via efficient memory exposure reduction and randomization,” in *Proc. of ASPLOS*, pp. 987–1000, 2020.
- [65] G. S. Kc, A. D. Keromytis, and V. Prevelakis, “Countering code-injection attacks with instruction-set randomization,” in *Proc. of CCS*, pp. 272–280, 2003.